A Syntactical Approach to Weak (Bi-)Simulation for Coalgebras

Jan Rothe [1,2]

Institut für Theoretische Informatik
TU Dresden
D-01062 Dresden, Germany.

Abstract

In [19] Rutten introduced the notion of weak bisimulations and weak bisimilarity for coalgebras of the functor $F(X) = X + O$. In the present paper I will introduce a notion of weak bisimulation for coalgebras based on the syntax of their functors for a large class of functors. I will show that my definition does not only coincide with the definition from [19], but with the definition for labelled transition systems as well. The approach includes a definition of weak bisimulation for Kripke structures, which might be of interest in its own right.

1 Introduction

As a result of considering process types as dual to data types, coalgebras gained attention from theoretical computer scientists. In this context, coalgebras are now also accepted as a semantics for classes in object oriented programming and specification, see [12] for the initial paper. Recently, there was a lot of work centred around the mentioned duality, especially to establish coalgebraic versions of Birkhoffs variety theorem, see e.g. [6], [2],[1].

It is commonly seen as one of the biggest advantages of coalgebras that they deliver standard notions like bisimulation, observational equality and (path-wise) modal operators "for free". Also, many people see coalgebras as generalisations of transition systems. However, some parts of this generalisation are still missing. When considering processes as transition systems notions like weak bisimulation and convergence are natural (see [9]) but it is not obvious how to introduce it to coalgebras.

In [19] Rutten proposes a notion of weak bisimulation for coalgebras of the functor $FX = X + O$, which are automata with terminal output, also

[1] Supported by a grant from the DFG, within the postgraduate programme GK 334.
[2] Email: janr@tcs.inf.tu-dresden.de

known as Elgot Machines. He uses this notion to establish denotational and operational semantics of while programs. In this paper I will propose a way of defining weak bisimulation for coalgebras for a large class of functors. This definition is inspired by [19] and from the context of transition systems.

The paper has the following structure: In the next section I will recapitulate the definition of coalgebras, labelled transition systems, and Kripke structures and the notions of weak bisimulation and bisimulation for transition systems together with some simple results. The third section establishes a translation from coalgebras to transition systems. After that the reader shall find the definition of weak bisimulation for coalgebras via that translation and a direct definition together with a result that a weak coalgebra bisimulation is indeed a weak transition system bisimulation. Section 5 contains a small case study that shows where the definitions can be usefully applied.

Along the lines of the general theory, you will find two examples. Firstly, transition systems themselves are considered as coalgebras, and it will be shown how the definitions behave for those. Secondly, Elgot Machines are considered to show that my approach generalises the one from [19].

2 Foundations

In this section you will find standard definitions that I will use throughout this paper. For more detailed presentations, refer to the literature, e.g [5],[18] for coalgebras and [3], [22] for transition systems.

Definition 2.1 [Coalgebra] Consider an endofunctor F on the category of sets and total functions **Set**. A coalgebra is a function $c : X \to F(X)$.

Example 2.2 [Functors] The following enumeration contains some special functors or ways to build functors from other functors. It only describes how the functors act on the sets - what they do with functions is then straightforward.

(i) $F : X \mapsto A$ is a constant functor.

(ii) $F : X \mapsto X$ is the identity functor.

(iii) $F : X \mapsto \mathcal{P}(X)$ is the covariant power-set functor.

(iv) $F : X \mapsto F_1(X) \times F_2(X)$ is the product of two functors F_1 and F_2.

(v) $F : X \mapsto F_1(X) + F_2(X)$ is the coproduct (disjoint union) of functors F_1 and F_2.

(vi) $F : X \mapsto F_1(X)^A$ is the constant exponent of some functor F_1 wrt. a set A.

As usual, associated with the product there are the right and left projection functions π_1 and π_2, the coproduct comes with the injections κ_1 and κ_2.

In the following I will consider the class of functors that is the least class containing (i), (ii) and (iii) which is closed under the constructions (iv)-(vi).

In the remaining part of this section I will turn my attention to labelled transition systems.

Definition 2.3 [Labelled Transition Systems] A *labelled transition system* (LTS) is a triple $Q = (X, L, R)$, where X is a set of states, L a set of labels, and $R \subseteq X \times L \times X$ is a transition relation.

Alternatively to the notation $(x, l, y) \in R$, I will write $x \xrightarrow{l} y$. There may be a distinguished label τ in the set of labels of an LTS. I will call this label *hidden label* and a such labelled transition *hidden transition*. It corresponds to an invisible (internal) action of the process that is modelled by the transition system. The non-hidden labels will be called visible or observable. I shall write $x \Rightarrow y$ if y is reachable from x within an arbitrary number of τ transitions. Formally, $\Rightarrow$ is the reflexive transitive closure of $\xrightarrow{\tau}$. Given a word $w = l_1...l_n \in L^*$, I will write $\overset{w}{\Rightarrow}$ for $\Rightarrow\xrightarrow{l_1}\Rightarrow ... \Rightarrow\xrightarrow{l_n}\Rightarrow$ and $\overset{w}{\rightarrow}$ for $\xrightarrow{l_1} ... \xrightarrow{l_n}$.

Example 2.4 [LTS as coalgebra] Consider a LTS $T = (X, L, R)$. Then T is a coalgebra $c : X \to \mathcal{P}(X)^L$, with $x' \in c(x)(l)$ if and only if $x \xrightarrow{l} x'$.

Definition 2.5 [Labelled Transition Systems with Attributes] A *labelled transition system with attributes* (LTSA) is a 5-tuple $Q = (X, A, L, v, R)$, where (X, L, R) is a LTS, A is a set of attributes, and $v \subseteq A \times X$ a relation that evaluates the attributes wrt. states.

In the following I will denote the fact that an attribute $a \in A$ holds in a state x by $x{\uparrow}a$, abbreviating $(a, x) \in v$, if v is clear from the context.

Remark 2.6 Obviously, a labelled transition system with attributes can be translated into a labelled transition system without attributes (LTS). There are many ways to do this. Since I want to define weak bisimulations here, let us follow the motivation of Hennessy and Milner from [3], where they defined weak bisimulation for transition systems.

For them, every possible observation is a transition and, as such, may change the state of the system. In our case, attribute observations are explicitly meant *not* to change the state of the system. Hence, these explicit observations by attributes should be considered as transitions that *do not* change the state of the system.

The resulting translation then takes a labelled transition system with attributes $T = (X, A, L, v, R)$ and transforms it into a labelled transition system $\mathsf{lts}(T) = (X, L + A, R')$, where for $l \in L$, $(x, \kappa_1 l, y) \in R'$ iff $(x, l, y) \in R$ and for all $x \in X$ and $a \in A$, we have $(x, \kappa_2 a, x) \in R'$ iff $x{\uparrow}a$. So we keep the complete transition structure of the original LTSA and add transition labelled by a from a state x to itself if a is an explicit observation in the state x.

Labelled transition systems with attributes are Kripke models from Modal Logics: A LTSA is a Kripke model for a multi-modal logic where the modalities are the labels from L and where the set of propositional variables is the set of

attributes A of the LTSA. So the translation from above is in fact a translation from Kripke models to labelled transition systems.

Definition 2.7 [Bisimulation for Transition Systems] Let $P = (X, L, R)$ and $Q = (Y, L, R')$ be LTS. A relation $S \subseteq X \times Y$ is a simulation from P to Q, if for all $(x, y) \in S$ and all $l \in L$ and $x' \in X$, $x \xrightarrow{l} x'$ implies there is a $y' \in Y$ such that $y \xrightarrow{l} y'$ and $(x', y') \in S$.

From a LTSA $T = (X, A, L, v, R)$ to $U = (Y, A, L, v', R')$, S is a simulation, if it is a simulation for the transition systems (X, L, R) and (Y, L, R'), and additionally it holds that $(x, y) \in S$ implies that for all $a \in A$, $x{\uparrow}a$ (in T) implies $y{\uparrow}a$ (in U).

A relation S is a bisimulation for a transition system (with attributes), if it is a simulation and its converse S^{-1} is a simulation. Two states $x \in X$ and $y \in Y$, are called (bi)similar, if there is a (bi)simulation containing the pair (x, y).

Obviously, the definition of bisimulation for LTSA also respects the translation into a LTS from Remark 2.6. That is, a relation S is a bisimulation for a LTSA T iff it is a bisimulation for $\mathsf{lts}(T)$. Observe also that Definition 2.7 is equivalent to the definition of bisimulation for Kripke models from modal logics.

Definition 2.8 [Weak Bisimulation for Labelled Transition Systems] Let $P = (X, L, R)$ and $Q = (Y, L, R')$ be LTS. The essence $\hat{w}$ of a word $w \in L^*$ is obtained from the word w by removing all occurrences of τ. A relation $S \subseteq X \times Y$ is a weak simulation from P to Q, if for all $(x, y) \in S$, $w \in L^*$, and $x' \in X$: If $x \stackrel{w}{\Rightarrow} x'$ then there is a $y' \in Y$, such that $y \stackrel{\hat{w}}{\Rightarrow} y'$ and $(x', y') \in S$.

A simulation S is a weak bisimulation, if additionally its converse S^{-1} is a weak simulation.

As there is no standard definition of weak bisimulation for LTSA, one has to decide how to deal with their attributes. As mentioned above, from the motivation of Hennessy and Milner in [3] it is natural to consider the attributes as special observations that *do not* change the state of the system.

I will pursue this idea in the following to define weak bisimulation for LTSA.

Definition 2.9 Let T and U a labelled transition systems with attributes as above. A relation $S \subseteq X \times Y$ is a *weak simulation* if it is a weak simulation for the (X, L, R) and (Y, L, R'), and additionally for all $(x, y) \in S$ and $a \in A$:

$$\forall x', x'' \in X. x \Rightarrow x' \wedge \ x'{\uparrow}a \ \wedge x' \Rightarrow x'' \text{ implies}$$
$$\exists y', y'' \in Y. y \Rightarrow y' \wedge \ y'{\uparrow}a \ \wedge y' \Rightarrow y'' \ \wedge \ (x'', y'') \in S$$

A weak simulation S is a *weak bisimulation* if S^{-1} is also a weak simulation.

The previous definition is also a definition of weak bisimulation for Kripke structures. This definition is new and might be of interest to people from the modal logics community.

For two labelled transition systems (with or without attributes) P and Q and two of their respective states x and y, I will write that x is (weakly) (bi)similar to y, if there is a (weak) (bi)simulation from P to Q containing (x, y), respectively. Further, I will write that Q simulates (is bisimilar to) P if there is a (bi)simulation from P to Q that relates every state from P to some state of Q (and every state in Q is related to some state in P for the bisimulation case).

As it is often easier to use alternative but equivalent formulations of weak (bi)simulations, consider the following

Lemma 2.10 *Any relation $S \subseteq X \times Y$ is a weak simulation from $P = (X, L, R)$ to $Q = (Y, L, R')$, if and only if the following statements hold for all $(x, y) \in S$:*

(i) *for all $l \in L$ with $l \neq \tau$ and for $x' \in X$ with $x \Rightarrow \stackrel{l}{\rightarrow} \Rightarrow x'$, there is a $y' \in Y$, such that $y \Rightarrow \stackrel{l}{\rightarrow} \Rightarrow y'$, and $(x', y') \in S$*

(ii) *for $x' \in X$, with $x \Rightarrow x'$, there is $y' \in Y$, with $y \Rightarrow y'$ and $(x', y') \in S$*

Proof. For the only if part, assume S is a weak simulation. Condition (i) trivially holds and (ii) follows from the definition of weak bisimulation for the empty word $w = \varepsilon$.

For the if part assume (i) and (ii) hold. For the condition on words, consider the following induction argument: The base case $w = \varepsilon$ is covered by (ii). For the inductive case assume we showed that for the word w_1 the weak simulation property holds (induction hypothesis). It remains to show that for the word $w = w_1 l$ for every $l \in L$ the property holds. Consider $(x, y) \in S$. By induction hypothesis, $x \stackrel{w_1}{\Longrightarrow} x'$, implies that there is a y', such that $y \stackrel{\hat{w_1}}{\Longrightarrow} y'$ and $(x', y') \in S$. From (i) conclude that if $x' \Rightarrow \stackrel{l}{\rightarrow} \Rightarrow x''$, then there is y'', such that $y' \Rightarrow \stackrel{l}{\rightarrow} \Rightarrow y''$ for $l \neq \tau$ and for $l = \tau$, $x' \Rightarrow \stackrel{\tau}{\rightarrow} \Rightarrow x''$ implies there is a y'' such that $y' \Rightarrow y''$. In any case $(x'', y'') \in S$. So one concludes that $x \stackrel{w_1 l}{\Longrightarrow} x''$ implies there is a y'', such that $y \stackrel{\hat{w_1 l}}{\Longrightarrow} y''$ and $(x'', y'') \in S$ for $l \neq \tau$ and for $l = \tau$ there is a y'', such that $y \stackrel{\hat{w_1}}{\Longrightarrow} y''$ and $(x'', y'') \in S$. By definition of $\widehat{w_1 l}$, this concludes the proof. □

The following easy lemma relates simulations for LTSA to simulations for LTS.

Lemma 2.11 *Let $T = (X, A, L, v, R_T)$ and $U = (Y, A, L, v', R_U)$ be LTSA and $P = \mathsf{lts}(T) = (X, L + A, R'_T)$ and $Q = \mathsf{lts}(U) = (Y, L + A, R'_U)$ be the corresponding LTS as given by Remark 2.6. Any simulation S from T to U is also a simulation from P to Q.*

Proof. Consider a simulation S from T to U. To check that S is a simulation

for Q, it suffices to show that S fulfils (i) and (ii) from Lemma 2.10. It is easy to see that (ii) holds, since R' contains R and Q has no additional hidden labels compared to T. For the same reason, condition (i) holds for the visible labels from L. For the new labels from A, (i) holds by definition of weak simulation for the LTSA T. □

There is another obvious observation that relates weak bisimulations with bisimulations for transition systems: It is possible to translate a LTS in such a way that a weak bisimulation for the original LTS is a bisimulation for the resulting one. The idea is to collapse series of τ-actions into one τ-action and to replace $\stackrel{l}{\Rightarrow}$ by $\stackrel{l}{\rightarrow}$. Van Glabbeek used a similar but slightly different construction in [21] to obtain a notion of (weak) bisimilarity that is a congruence.

Definition 2.12 For a labelled transition system $P = (X, L, R)$, define the *essence* of the transition relation $\overline{R}$ by

(i) $(x, l, x') \in \overline{R}$ iff $x \stackrel{l}{\Rightarrow} x'$ for all $l \neq \tau \in L$ and

(ii) $(x, \tau, x') \in \overline{R}$ iff $x \Rightarrow x'$.

The essence of the transition system P is $\overline{P} = (X, L, \overline{R})$

Lemma 2.13 *Consider two LTS $P = (X, L, R)$ and $Q = (Y, L, R')$ and their essences $\overline{P}$ and $\overline{Q}$ as defined above. $S \subseteq X \times Y$ is a weak (bi)simulation from P to Q if and only if it is a (bi)simulation from $\overline{P}$ to $\overline{Q}$.*

Proof. For this proof it suffices to consider only weak simulation and simulation. The weak bisimulation vs. bisimulation case is then straightforward symmetry. Unless stated otherwise, $\rightarrow$ refers to transitions in $\overline{(-)}$ and $\Rightarrow$ refers to series of transitions in $(-)$, for P and Q, respectively.

For the "only if"-direction, let S be a weak simulation for the original LTS. Consider $(x, y) \in S$. By the definition of simulation, it remains to show that for all $l \in L$, $x \stackrel{l}{\rightarrow} x'$ implies that there is a y' such that $y \stackrel{l}{\rightarrow} y'$ with $(x', y') \in S$. For $l \neq \tau$ compute:

$$\begin{aligned} x \stackrel{l}{\rightarrow} x' &\iff x \stackrel{l}{\Rightarrow} x' \\ \text{S is weak sim.} &\implies \exists y'.y \stackrel{l}{\Rightarrow} y' \text{ and } (x', y') \in S \\ &\iff \exists y'.y \stackrel{l}{\rightarrow} y' \text{ and } (x', y') \in S \end{aligned}$$

For $l = \tau$, we get the similar

$$\begin{aligned} x \stackrel{\tau}{\rightarrow} x' &\iff x \Rightarrow x' \\ \text{S is weak sim.} &\implies \exists y'.y \Rightarrow y' \text{ and } (x', y') \in S \\ &\iff \exists y'.y \stackrel{\tau}{\rightarrow} y' \text{ and } (x', y') \in S \end{aligned}$$

Hence, S is a simulation for the essences.

For the "if"-direction consider a simulation S for the essences and a pair $(x, y) \in S$. As above compute for $l \neq \tau$:

$$\begin{aligned} x \stackrel{l}{\Rightarrow} x' &\iff x \stackrel{l}{\rightarrow} x' \\ \text{S is simulation} &\implies \exists y'.y \stackrel{l}{\rightarrow} y' \text{ and } (x', y') \in S \\ &\iff \exists y'.y \stackrel{l}{\Rightarrow} y' \text{ and } (x', y') \in S \end{aligned}$$

and for τ-transitions:

$$\begin{aligned} x \Rightarrow x' &\iff x \stackrel{\tau}{\rightarrow} x' \\ \text{S is simulation} &\implies \exists y'.y \stackrel{\tau}{\rightarrow} y' \text{ and } (x', y') \in S \\ &\iff \exists y'.y \Rightarrow y' \text{ and } (x', y') \in S \end{aligned}$$

Hence, S is weak simulation for the original LTS. □

The lemma above can be extended to LTSAs: The attributes of a state in the resulting LTSA are the attributes of all states that are $\Rightarrow$-accessible in the originating LTSA:

Definition 2.14 [Essence of a LTSA] For a labelled transition system with attributes $T = (X, A, L, v, R)$, define the essence $\overline{v}$ of the observations by $(a, x) \in \overline{v}$ if and only if there is a x', such that $x \Rightarrow x'$ and $(a, x') \in v$. The essence $\overline{T}$ of T is then $(X, A, L, \overline{v}, \overline{R})$

Lemma 2.15 *Consider two LTSAs T and U as above and their essences $\overline{T}$ and $\overline{U}$. A relation $S \subseteq X \times Y$ is a weak (bi)simulation for T if and only if it is a (bi)simulation for $\overline{T}$.*

Proof. Straightforward along the lines of the proof for Lemma 2.13 and by definition of $\overline{v}$. □

3 From Coalgebras to Transition Systems

In this section I will develop a translation from coalgebras $c : X \to F(X)$ to transition systems over the same state space X. It is obvious that possible (coalgebraic) observations about a state x must be captured by attributes of x and successor states of x via the coalgebra raise transitions in the resulting LTSA.

The first part of the section contains the machinery of the translation for coalgebras over structured functors. Most of the presentation is based on work of Kurz [7], Rößiger [13], and Jacobs [4]. The contribution of this section is a novel definition of attributes of a coalgebra. For a future semantical approach to weak bisimulation, the work of Pattinson about semantical principles in the modal logics for coalgebras [11] should provide the needed tools.

I start with the well-known parts: The definition of transitions. This is done by induction over the structure of the functor - transition labels are paths through the syntax tree of the functor.

Definition 3.1 [Labels of a functor, Successor via the coalgebra] Let F be a functor. Define the set of transition labels L_F, and the corresponding successor relation $\succ_F \subseteq F(X) \times L_F \times X$ by induction over the structure of F:

$$
\begin{array}{lll}
F(X) = X & : L_F = \mathbf{1} = \{*\} & \succ_F = \{(x, *, x) | x \in X\} \\
F(X) = A & : L_F = \emptyset & \succ_F = \emptyset \\
F(X) = \mathcal{P}(X) & : L_F = \mathbf{1} & \succ_F = \{(S \subseteq X, *, x) | x \in S\} \\
F(X) = F_1(X) \times F_2(X) & : L_F = L_{F_1} + L_{F_2} & \succ_F = \{((f_1, f_2), \kappa_i l, x) | \succ_{F_i} (f_i, l, x)\} \\
F(X) = F_1(X) + F_2(X) & : L_F = L_{F_1} + L_{F_2} & \succ_F = \{(\kappa_i f_i, \kappa_i l, x) | \succ_{F_i} (f_i, l, x)\} \\
F(X) = F_1(X)^A & : L_F = A \times L_{F_1} & \succ_F = \{(f, (a, l_1), x) | \succ_{F_1} (f(a), l_1, x\}
\end{array}
$$

For a translation from coalgebras into LTSAs, the *right* definition of attributes of a functor is still missing. Unfortunately, the standard approach for strong bisimulation approaches to take (subsets of) $F\mathbf{1}$ does not work here.

Example 3.2 The main problem in the $F\mathbf{1}$-as-attributes approach is that the possibility to perform transitions is observable as an attribute. Considering transition systems as coalgebras for the functor $F(X) = \mathcal{P}(X)^L$, one gets subsets of L as possible observations. Although any subset of L not containing the hidden transition τ would not pose problems to the idea of τ as non-observable transition, subsets of L containing τ are allowed as attributes in one approach (see [10]). Even worse the full set L is the set of observations for that functor in [14]. The same argument applies to the functor $F(X) = X + O$, where $F\mathbf{1}$ makes the possibility of a transition visible. Thus the known definitions of observations contradict the idea of hiding τ-transitions.

The idea to overcome the problem, is to make all transitions invisible by attributes – henceforth, all transitions will only be visible as transitions, not as attributes. The first obvious idea to consider subsets of $F(\emptyset)$ fails. Take for instance a coalgebra for the functor $F(X) = X \times A + X$. One might choose that the possible transition introduced by the second injection shall be invisible. Still then it should in general be possible to observe elements of A. But $F(\emptyset) = \emptyset$, so no attributes would belong to the functor.

The final solution is to replace all products ($\times$) in the functor by sums ($+$). The next definition makes this explicit.

Definition 3.3 [Attributes for a functor] Define the set A_F of attributes of a functor F inductively over the structure of the functor and at the same time, define the relation $v_F \subseteq F(X) \times A_F$:

$$
\begin{array}{lll}
F(X) = X & : A_F = \emptyset & \\
F(X) = A & : A_F = A & : v_F(f, a) \iff f = a \\
F(X) = \mathcal{P}(X) & : A_F = \emptyset & \\
F(X) = F_1(X) \times F_2(X) & : A_F = A_{F_1} + A_{F_2} & : v_F((f_1, f_2), \kappa_i a) \iff v_{F_i}(f_i, a) \\
F(X) = F_1(X) + F_2(X) & : A_F = A_{F_1} + A_{F_2} & \\
 & : v_F(\kappa_i f_i, \kappa_j a) \iff i = j \wedge v_{F_i}(f_i, a) & \\
F(X) = F_1(X)^A & : A_F = A \times A_{F_1} & \\
 & : v_F(f : A \to F_1(X), (a, a')) \iff v_{F_1}(f(a), a') &
\end{array}
$$

Example 3.4 For the transition systems functor $F_1(X) = \mathcal{P}(X)^L$, the set of attributes is the set $A_{F_1} = \emptyset \times L = \emptyset$, no explicit observations are possible. For the functor $F_2(X) = X + O$, $A_{F_2} = \emptyset + O = O$. Hence all elements of the termination set O can be observed.

When specifying a system, it is often useful to have internal attributes that shall not be visible from outside the system. To allow this kind of restrictions, more freedom in the choice of attributes is necessary.

Definition 3.5 [LTSA for a coalgebra] Let $c : X \to F(X)$ be a coalgebra. Let $T \subseteq L_F$ be a set of labels of the functor F that will be considered as hidden transitions and let $O \subseteq A_F$ be a set of attributes for the functor F. Then $\mathsf{ltsa}_T^O(c) = (X, O, L, v, R)$ is the labelled transition system with attributes for the coalgebra c, if $L = L_F \setminus T \cup \{\tau\}$ and for all $x \in X$ the following hold:

(i) for all $a \in O$, $x{\uparrow}a$ if and only if $v_F(c(x), a)$, for v_F as in Definition 3.3,
(ii) for all $l \in L_F \setminus T$, $x \xrightarrow{l} y$ if and only if $\succ_F (c(x), l, y)$, and
(iii) $x \xrightarrow{\tau} y$ if and only if there exists $l \in T$, such that $\succ_F (c(x), l, y)$.

T and O can be omitted in ltsa_T^O if they are clear from the context.

Basically, the definition above takes a set of hidden transitions of a coalgebra (a subset of the labels of the functor) and a set of attributes and computes the resulting LTSA, comprising the transition and attribute structure of the coalgebra. Hidden transitions are marked with τ and hidden attributes do not appear in the resulting LTSA.

Example 3.6 [LTS as Coalgebras] Consider a labelled transition system $P = (X, L, R)$ with hidden label τ as coalgebra as in Example 2.4. The set of labels for this functor is $L_F = L$, the set of attributes is $A_F = \emptyset$. Now I choose to hide the hidden labels from the original LTS P: $L_P = \{\tau\} \subseteq L_F$. The set of observations O can only be the empty set. The resulting LTSA $\mathsf{ltsa}(c) = (X, O, L, v, R)$ is the original LTS itself.

Example 3.7 [Elgot Machine] A coalgebra $c : X \to X + O$ where the only possible state transition $\kappa_1 *$ is considered as hidden transition gets translated into a LTSA where every state has either exactly one τ-successor or permits an observation $o \in O$. A successor is the result of the coalgebra applied to a state, if the result is from the first injection to $X + O$. A direct observation can be made in a state, if the result of applying the coalgebra comes from the second injection to $X + O$.

4 Weak Bisimulations

The translation that I introduced in the previous section allows the definition of weak bisimulation for coalgebras. At first, I will define it via translation to transition systems.

Definition 4.1 [Weak Bisimulation 1] Let F be a functor with L_F as set of labels and A_F as set of attributes as above and $c : X \to F(X)$ and $d : Y \to F(Y)$ be coalgebras. For a set of hidden labels $T \subseteq L_F$ and a set of permitted observations $O \subseteq A_F$, a relation $S \subseteq X \times Y$ is a weak (bi)simulation, if and only if S is a weak (bi)simulation for the underlying transition system with attributes $\mathsf{ltsa}(c)$.

Proposition 4.2 (LTS) *Assume labelled transition systems T, U, and their coalgebraic version $c : X \to \mathcal{P}(X)^L$ and $d : Y \to \mathcal{P}(Y)^L$, respectively. Then a relation $R \subseteq X \times Y$ is a weak bisimulation betweem T and U if and only if it is a weak bisimulation between c adn d.*

Proof. As seen in Example 3.6, a labelled transition system as a coalgebra is translated via $\mathsf{ltsa}(c)$ into itself. Hence, the resulting notion of weak bisimulation betwen coalgebraic LTS and the notion of weak bisimulation for labelled transition systems coincide. □

Proposition 4.3 *Consider two Elgot machines $c : X \to X + O$ and $d : Y \to Y + O$. Then the notion of weak bisimulation from Definition 4.1 coincides with the definition from [19].*

Proof. Consider the translation of the Elgot machines into LTSAs as in Example 3.7. Using the notation $x \to x'$ for the fact that $c(x) = \kappa_1 x'$ or $d(x) = \kappa_1 x'$ (hence, $\to$ is a relation on $X \times X$), and for $\Rightarrow$ being the reflexive transitive closure of $\to$, one gets the following:

$S \subseteq X \times Y$ is a weak bisimulation, iff for all $(x, y) \in S$, the following hold:

(i) $x \Rightarrow x'$ implies $\exists y'.y \Rightarrow y'$ and $(x', y') \in S$.
(ii) $x \Rightarrow x'$ and $c(x') = \kappa_2 o$ implies $\exists y'.y \Rightarrow y'$ and $c(y') = \kappa_2 o$ and $(x', y') \in S$.
(iii) $y \Rightarrow y'$ implies $\exists x'.x \Rightarrow x'$ and $(x', y') \in S$.
(iv) $y \Rightarrow y'$ and $c(y') = \kappa_2 o$ implies $\exists x'.x \Rightarrow x'$ and $c(x') = \kappa_2 o$ and $(x', y') \in S$.

An easy induction over the length of $\Rightarrow$-chains shows that this definition is equivalent to that in Ruttens paper. □

Obviously, the approach of translating a coalgebra into a transition system and then looking at the definition of weak bisimulation, is not the way one wants to go when working with real specifications. For this reason, I will give a direct definition in terms of the considered coalgebra.

First, I define a successor relation and an observation relation for coalgebras. Both shall exactly correspond to their transition system counterparts.

Definition 4.4 For a coalgebra $c : X \to F(X)$, where L_F is the set of labels and A_F is the set of possible attributes for the functor F, define the successor relation $(-) \overset{(-)}{\rightsquigarrow}_c (-) \subseteq X \times L_F \times X$ by $x \overset{l}{\rightsquigarrow}_c y \iff \succ_F (c(x), l, y)$ and the observation relation $(-)\uparrow_c(-) \subseteq X \times A_F$ by $x \uparrow_c a \iff v_F(c(x), a)$.

Example 4.5 For a LTS-coalgebra $c : X \to \mathcal{P}(X)^L$, we get that $x \overset{l}{\rightsquigarrow}_c y$ holds iff $x \overset{l}{\to} y$ holds in the corresponding LTS.

For an Elgot Machine $c : X \to X + O$, $x \overset{l}{\rightsquigarrow}_c y$ holds iff $l = \kappa_1 *$, which is the only possible label, and $c(x) = \kappa_1 y$. As observation relation, we get $x{\uparrow_c}a$ if and only if $c(x) = \kappa_2 a$.

To ease the presentation, I introduce the following helper notion: $x \Longrightarrow_c^T y \subseteq X \times X$ is an abbreviation for $\exists w = l_1 \ldots l_n \in T^*.x \overset{l_1}{\rightsquigarrow}_c x_1 \ldots x_{n-1} \overset{l_n}{\rightsquigarrow}_c y$, the closure of all labelled transitions for a set of (hidden) labels T.

Definition 4.6 [Weak Bisimulation 2] Let $c : X \to F(X)$ and $d : Y \to F(Y)$ be coalgebras, L_F be the set of labels of F and $T \subseteq L_F$ be the chosen set of hidden labels. Let A_F be the set of attributes for F and $O \subseteq A_F$ be the chosen set of observations. Then a relation $S \subseteq X \times Y$ is a weak bisimulation wrt. T and O, iff for all $(x, y) \in S$ and all $l \in L_F \setminus T$ and all $a \in O$, the following hold:

(i) $x \Longrightarrow_c^T \overset{l}{\rightsquigarrow}_c \Longrightarrow_c^T x'$ implies $\exists y'.y \Longrightarrow_c^T \overset{l}{\rightsquigarrow}_c \Longrightarrow_c^T y'$ and $(x', y') \in S$.
(ii) $x \Longrightarrow_c^T x'$ implies $\exists y'.y \Longrightarrow_c^T y'$ and $(x', y') \in S$.
(iii) $x \Longrightarrow_c^T x' \Longrightarrow_c^T x''$ and $x'{\uparrow_c}a$ implies $\exists y', y''.y \Longrightarrow_c^T y' \Longrightarrow_c^T y''$ and $y'{\uparrow_c}a$ and $(x'', y'') \in S$.
(iv) $y \Longrightarrow_c^T \overset{l}{\rightsquigarrow}_c \Longrightarrow_c^T y'$ implies $\exists x'.x \Longrightarrow_c^T \overset{l}{\rightsquigarrow}_c \Longrightarrow_c^T x'$ and $(x', y') \in S$.
(v) $y \Longrightarrow_c^T y'$ implies $\exists x'.x \Longrightarrow_c^T x'$ and $(x', y') \in S$.
(vi) $y \Longrightarrow_c^T y' \Longrightarrow_c^T y''$ and $y'{\uparrow_c}a$ implies $\exists x', x''.x \Longrightarrow_c^T x' \Longrightarrow_c^T x''$ and $x'{\uparrow_c}a$ and $(x'', y'') \in S$.

For S being a weak simulation, S must fulfil (i)-(iii).

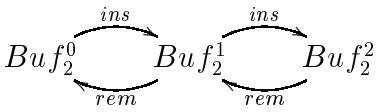
Theorem 4.7 (Equivalence) *Definition 4.1 and Definition 4.6 coincide.*

Proof. [Sketch] Consider coalgebras c and d and L_F, T, A_F and O as above. First, it is easy to see that $x \overset{l}{\rightsquigarrow}_{(-)} x'$ iff $x \overset{l}{\to} x'$ in $\mathsf{ltsa}(-)$. By induction on the length of the chain of transitions, it follows that $x \Longrightarrow_{(-)}^T x'$ if and only if $x \Rightarrow x'$ in $\mathsf{ltsa}(-)$. It is then easy to see, that every relation S that fulfils Definition 4.1 also fulfils 4.6 and vice versa. □

5 Buffers

A very basic example of weak bisimulation is the specification of a 2-Buffer, a queue that can store at most 2 elements. Using transition systems, there are at least two ways to do that. The first, straight-forward way to do so is

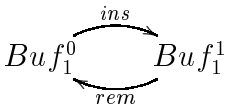
$$
\begin{aligned}
Buf_2^0 &= ins.Buf_2^1 \\
Buf_2^1 &= ins.Buf_2^2 + rem.Buf_2^0 \\
Buf_2^2 &= rem.Buf_2^1
\end{aligned}
$$



where the left part specifies the behaviour in some process-algebraic equations and the right part shows the generated transition structure.

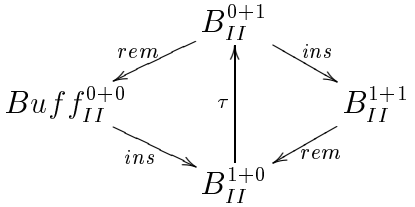
Another way to do it uses a prior definition of a 1-Buffer, renaming, and communication. First specify a 1-Buffer:

$$Buf_1^0 = ins.Buf_1^1$$
$$Buf_1^1 = rem.Buf_1^0$$



Then take two of them, rename the rem-operation of the first one to int and the ins of the second one to $\overline{int}$ and make int an internal action:

$$Buf_{II}^{0+0} = new\ int.(\ \{rem := int\}Buf_1^0 |\ \{ins := \overline{int}\}Buf_1^0)$$



It is straightforward to show that Buf_{II}^{0+0} is weakly bisimilar to Buf_2^0.

In my coalgebraic specification, I want to model more than just inserting and removing: it should be possible to insert and remove elements of some set A. First, let us consider the interface of the Buffer. Obviously, one needs two functions that perform state transitions - a insert function ins and a remove operation rem. The insert function shall take an additional argument $a \in A$ that is to be inserted into the buffer. To model the possibility of ins to fail, I will use an additional error element, so insert has the type $ins : X \to X^A + \mathbf{1}$. The interface functor of ins is $F_{ins}(X) = X^A + 1$. In case the buffer is nonempty, rem shall deliver a successor state together with the element that was removed, and an error element will be returned if the buffer is empty. Hence rem has the type $X \to A \times X + \mathbf{1}$ with $F_{rem}(X) = A \times X + \mathbf{1}$. The resulting interface functor for both functions together is then $F_{\langle ins,rem\rangle}(X) = (X^A + 1) \times (A \times X + \mathbf{1})$.

When specifying a 2-Buffer in a direct (observational) way, as shown in the first transition system, then this interface is enough. However, when one wants to specify a 2-Buffer as containing at most two elements of A, then one additionally needs two attributes $b_1, b_2 : X \to A + 1$ and a operation $push : X \to X+1$. The method $push$ shall model the communication between those two internal attributes, moving the content of b_1 into b_2 if b_2 is empty.

My goal is to relate coalgebras for both mentioned buffer specifications. Since my definition allows to relate only coalgebras of the same functor, both must have the same interface functor. Hence, for 2-Buffers I have $F_{buf}(X) = (X^A + \mathbf{1}) \times (A \times X + \mathbf{1}) \times (X + \mathbf{1}) \times (A + \mathbf{1}) \times (A + \mathbf{1})$. For a coalgebra $c : X \to F_{buf}(X)$ the operations ins, rem, $push$, b_1, and b_2 are abbreviations for $\pi_1 \circ c$, ..., $\pi_5 \circ c$ respectively.

First, consider a specification which I would like to call observational. A coalgebra $c : X \to F_{buf}(X)$ is a Buf_2-coalgebra iff for all $x \in X$ and all

$a, b, c \in A$ the following hold:

(i) If the buffer is empty, i.e. $rem(x) = *$, then ins is possible, i.e. there is a successor state y with $ins(x)(a) = y$ and $rem(y) = (a, z)$ and x and z are (strongly!) bisimilar. This covers the case where the buffer is empty.

(ii) If the buffer is not empty, i.e. $rem(x) = (a, y)$ for some y, then
- If the buffer contains exactly one element of A, i.e. $rem(y) = *$, then it is possible to insert another element, i.e. there is a successor state z, such that $ins(x)(b) = z$, and from that resulting buffer one can remove the first element and obtain a successor state z' that shall act exactly like a buffer that contains only b. That means, $rem(z) = (a, z')$ and z' is bisimilar to the outcome of $ins(y)(b)$
- If the buffer contains two elements of A, i.e. $rem(y) = (b, z)$, then the insert operation is not possible, i.e. $ins(x) = *$.

(iii) There is a y such that $push(x) = y$ and x and y are bisimilar.

In the following I describe 2-Buffers in an implementational way: The idea is that b_1 and b_2 are internal and I will describe the outcome of the methods of the coalgebra in how they depend from and influences these attributes. A coalgebra $c : X \to F_{buf}(X)$ is a Buf_{II}-coalgebra iff for all x, y, z, z' and all $a, b, c \in A$ the following hold:

(i) If b_1 is empty in x, i.e. $b_1(x) = \kappa_2 *$, then $ins(x)(a) = \kappa_1 y$ and $b_1(y) = a$ and $b_2(y) = b_2(x)$. Furthermore, $push(x) = \kappa_2 *$.

(ii) If $b_2(x) = \kappa_2 *$, then $rem(x) = \kappa_2 *$ and $push(x) = \kappa_1 y$ and $b_2(y) = b_1(x)$.

(iii) If b_1 is not empty, i.e. $b_1(x) = \kappa_1 a$, then $ins(x)(b) = \kappa_2 *$.

(iv) if $b_2(x) = \kappa_1 a$, then $rem(x) = \kappa_1(a, y)$ and $b_2(y) = \kappa_2 *$ and $b_1(y) = b_1(x)$. Further, $push(x) = \kappa_2 *$

As announcedabove, in the specification all the operations work on the "internal" attributes, which is obviously closer to an implementation of 2-Buffers than the observational specification.

Using the definitions from the previous sections, there are the following labels for the functor: $L_{F_{buf}}(X) = \{l_i^a | a \in A\} \cup \{l_r, l_p\}$, where $l_i^a = \kappa_1(a, \kappa_1 *)$, $l_r = \kappa_2\kappa_1\kappa_2 *$, and $l_p = \kappa_3\kappa_1 *$, addressing the successor states via $ins(-)(a)$, $rem(-)$, and $push(-)$, respectively. The set of possible attributes is the sum of the attributes for ins: $\mathbf{1}$, rem: $A + \mathbf{1}$, $push$: $\mathbf{1}$, b_1 : $A + 1$ and b_2 : $A + 1$.

One will not consider the attributes b_1 and b_2 and the observations that results from $push$ as these are understood to be internal. The resulting set of observations is $O = \mathbf{1} + (A + \mathbf{1})$. For a coalgebra $c : X \to F_{buf}(X)$ and $x \in X$, we have that $x \uparrow_c \kappa_1 *$ if insert fails, $x \uparrow_c \kappa_2\kappa_1 a$ if rem succeeds, i.e. there exists $x' \in X$ such that $rem(x) = (a, x')$, and $x \uparrow_c \kappa_2\kappa_2 *$ if remove fails. Injections will be omitted whenever possible. The only element in the set of hidden labels shall be l_p.

Consider a Buf_2-coalgebra c and a Buf_{II}-coalgebra d. Then d weakly simulates c but *not* vice versa: construct a $S \subseteq X \times Y$ that is a simulation from c to d:

(i) If x models the empty buffer and y models the empty buffer then $(x, y) \in S$.
(ii) If x models a buffer containing exactly one element a and y models a buffer containing exactly the same element a, then $(x, y) \in S$.
(iii) If x is a buffer containing a and b, and y contains the same elements in the same order, then $(x, y) \in S$.
(iv) No other $x \in X$ and $y \in Y$ are related via S.

Assume that $(x, y) \in S$ and both are empty buffers. Then the hidden *push* does not change anything, all observations that can be made in x can be made in y, *rem* can not be successfully applied and *ins* delivers a successor state that models buffers containing exactly the same (one) element for the same input, hence those successor states are in S by (ii). Assume that $(x, y) \in S$ and both contain one element a. Then $x \uparrow_c a$. But $y \uparrow_d a$ does not always hold, since a could be in b_1, hence not readable for *rem*. Only after a (hidden) *push*-step, a will be in b_2. Hence $y \overset{l_p}{\rightsquigarrow}_d y' \uparrow_d a$. Also $(x, y') \in S$, since y' models a buffer containing only a, as well. The same argument applies to the *ins* and *rem* methods, each of which can be applied to y after at most one *push*-step, and they model the same buffers as the result of applying *ins* or *rem* to x, respectively. Hence the resulting states are in S, again. In case x and y contain two elements a and b in the same order, then the observations are immediately the same (without applying *push* to y). The only possible transition is *rem* and this transition leads in both cases to successor states modelling the same buffer. Hence, these successors must be in S.

Why are c and d not weakly bisimilar? Because the set O of observations allows to observe when a transition is *impossible*. So one can determine the difference between a state $x \in X$ and a state $y \in Y$, when they both model a buffer with one element, but y models the buffer, where this element still resides in the internal variable b_1. Then, *rem* is impossible in y, i.e. $y \uparrow_d \kappa_2\kappa_2*$. But no number of *push*-steps applied to x reaches a state, where *rem* fails. Hence, x and y can not be weakly bisimilar. The whole problem does not occur in the LTS-specification from the beginning of the section, since the definition of weak bisimulation for LTS, does not take the observation that an action can *not* occur into acount. The solution for the coalgebraic specifications here is easy - one has to restrict the set of observations even further: for $O = A$ and $x \uparrow_c a$ if $rem(x) = \kappa_1(x', a)$ for some x', the relation S constructed is indeed a bisimulation.

The buffer example raises two open problems: It is obvious, that the internal actions and attributes from the Buf_{II}-Specification are not needed in the Buf_2 specification at all. They only complicate the interface and the assertions. As a consequence, it would be nice to have a notion of weak (bi)simulation for coalgebras of different functors. The problem of finding a good set of observations for coalgebras became visible when considering weak simulation vs. weak bisimulation for the such sets.

6 Conclusions

It is well-known how to define bisimulation for coalgebras. This notion is not only of theoretical interest, it also serves as main part of the semantics for the class specification language CCSL (see [17], [15] and for an extensive presentation [20, Chapter 4]).

The notion of weak bisimulation as presented in this paper and a small case study give hope that weak bisimulation can be a tool in coalgebraic verification. It might also lead to new insights into coalgebraic refinement as weak bisimulation could provide means to compare coalgebras for different functors in the future.

There is a lot work to be done to get a deeper understanding of the field. The main weakness of the approach presented in this paper is the lack of a diagramatical, semantical definition of weak bisimulation. There was a solution for this in [19]. However, this solution does not scale to the whole class of functors that I consider in a straightforward way. This issue is subject of joint research with Dragan Mašulović. A first approach extending the definition from [19] can be found in [8], more possibly in [16].

Acknowledgements: I would like to thank Dragan Mašulović, Horst Reichel, and Hendrik Tews for many helpful comments on earlier versions of this paper.

References

[1] R. Goldblatt. A calculus of terms for coalgebras of polynomial functors. In A. Corradini, M. Lenisa, and U. Montanari, editors, *Coalgebraic Methods in Computer Science*, volume 44 of *ENTCS*, pages 160–183, April 2001.

[2] H.P. Gumm. Birkhoffs variety theorem for coalgebras. *Contributions to General Algebra*, 13:159–173, 2000.

[3] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, January 1985.

[4] B. Jacobs. The temporal logic of coalgebras via galois algebras. Technical Report CSI-R9906, Computing Science Institute, University of Nijmegen, 1999. To appear in Mathematical Structures in Computer Science.

[5] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:222–259, 1997.

[6] A. Kurz. A co-variety-theorem for modal logic. In *Proceedings of Advances in Modal Logic, Uppsala*. CSLI,Stanford, 1998. Revised Version.

[7] A. Kurz. Specifying coalgebras with modal logic. In B. Jacobs, L. Moss, H. Reichel, and J. Rutten, editors, *Proceedings of CMCS*, volume 11 of *ENTCS*, Lisbon, March 1998.

[8] D. Mašulović. Towards coalgebraic behaviourism. In L. Moss, editor, *Proceedings of CMCS*, volume 65.1 of *ENTCS*, 2002.

[9] R. A. Milner. *Communicating and Concurrency*. Prentice Hall, New York, 1989.

[10] D. Pattinson. Semantical principles in the modal logic of coalgebras. Technical report, Institut Informatik, LMU München, 2000.

[11] D. Pattinson. Semantical principles in the modal logic of coalgebras. In A. Ferreira and H. Reichel, editors, *Proceedings of STACS*, volume 2010 of *LNCS*. Springer, 2001.

[12] H. Reichel. An approach to object semantics based on terminal coalgebras. *Mathematical Structures in Computer Science*, 5:129–152, 1995.

[13] M. Rößiger. Languages for coalgebras on datafunctors. *ENTCS*, 19:55–76, 1999.

[14] M. Rößiger. *Coalgebras, Clone Theory and Modal Logic*. PhD thesis, Univ. of Dresden, Germany, 2000.

[15] J. Rothe. Modal logics for coalgebraic class specification. Master's thesis, Inst. Theor. Informatik, TU Dresden, D-01062 Dresden, Germany, 2000.

[16] J. Rothe and D. Mašulović. Towards weak bisimulation for coalgebras. In A. Kurz, editor, *Proceedings of CMCIM*, volume 68.1 of *ENTCS*, 2002. Submitted.

[17] J. Rothe, H. Tews, and B. Jacobs. The coalgebraic class specification language CCSL. *Journal of Universal Computer Science (JUCS)*, 7(2):175–193, 2001.

[18] J. Rutten. Universal coalgebra: a theory of systems. Report CS-R9652, Centrum voor Wiskunde en Informatica, CWI, P.O. Box 94079, 1090GB Amsterdam, The Netherlands, 1996.

[19] J. Rutten. A note on coinduction and weak bisimilarity for while programs. *Theoretical Informatics and Applications*, 33:393–400, 1999.

[20] H. Tews. *Coalgebraic Specification and Verification*. PhD thesis, Dresden University of Technology, 2002. Forthcomming.

[21] R.J. van Glabbeek. Bounded nondeterminism and the approximation induction principle in process algebra. In F.J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *Proceedings of STACS 1987*, number 247 in LNCS, pages 336–347. Springer, 1987. Extended Abstract.

[22] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 1–148. Oxford University Press, 1995.